# DECISION TREES FOR REAL-TIME TRANSIENT STABILITY PREDICTION

Steven Rovnyak * Student Member, IEEE
Stein Kretsinger ** non-member
James Thorp * Fellow, IEEE
Donald Brown ** Senior Member, IEEE

* Cornell University
Ithaca, New York

** University of Virginia
Charlottesville, Virginia

Keywords: Adaptive protection, decision trees, pattern recognition, phasor measurements, real-time, transient stability.

ABSTRACT - The ability to rapidly acquire synchronized phasor measurements from around the system opens up new possibilities for power system protection and control. This paper demonstrates how decision trees can be constructed off-line and then utilized on-line for predicting transient stability in real-time. Primary features of the method include building a single tree for all fault locations, using a short window of realistic-precision post-fault phasor measurements for the prediction, and testing robustness to variations in the operating point. Several candidate decision trees are tested on 40,800 faults from 50 randomly generated operating points on the New England 39 bus test system.

## 1. INTRODUCTION

With the advent of systems capable of making real-time phasor measurements, the real-time assessment of the stability of a transient event in the power system has become an important area of investigation. By synchronizing sampling of microprocessor based systems, phasor calculations can be placed on a common reference [1]. Commercially available systems based on GPS (Global Positioning System) satellite time transmissions can provide synchronization to 1 microsecond accuracy. The phasors obtained from a period or more of samples from all three phases provide a precise estimate of the positive sequence voltage phasor at a bus. The magnitudes and angles of these phasors comprise the state of the power system and are used in state estimation and transient stability analysis. By communicating time-tagged phasor measurements to a central location, the dynamic state of the system can be tracked in real time. Utility experience indicates that communication bandwidths can handle 12 complete sets of phasor measurements per second [2], which corresponds to one set every 5 cycles.

Using these phasor measurements for real-time transient stability prediction can advance the fields of protection and control. Out-of-step relaying is an obvious area of application. If an evolving swing could be determined to be stable or unstable, then the appropriate blocking or tripping could be initiated. A control application might involve determining whether the event would be stable under a variety of control options. In both cases the determination of stability or instability must be accomplished faster than real time in order for effective action to be taken. In other words it is necessary to predict the outcome before it actually occurs.

93 SM 530-6 PWRS A paper recommended and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1993 Summer Meeting, Vancouver, B.C., Canada, July 18-22, 1993. Manuscript submitted Jan. 4, 1992; made available for printing May 26, 1993.

PRINTED IN USA

Many transient stability assessment techniques while simple in off-line application are too complicated for real-time use. Real-time monitoring obviates the need for some of these techniques since the system itself is actually solving the differential equations. What is required is a computationally efficient way of processing the real-time measurements to determine whether an evolving event will ultimately be stable or unstable. Given the possible pay-off, the off-line computation can be extensive if the real-time speed is fast. The availability of powerful workstations and parallel supercomputing make new approaches to the problem possible.

Decision trees are a type of classifier that can be constructed off-line from a training set of examples [3,4]. In this paper, decision trees are used to classify a transient swing as either stable or unstable on the basis of real-time phasor measurements. Rather than attempting to solve the power system model in real-time, extensive simulations are performed off-line in order to capture the essential features of system behavior. The tree building process statistically analyzes this data and constructs a decision tree designed to correctly classify new, unseen examples. The resulting decision tree classifier is compact and extremely fast, thus well suited for on-line use.

Our decision tree approach falls into the broader category of pattern recognition. In the past 20 years, many forms of pattern recognition have been applied to the power system transient stability problem with varying degrees of success and sophistication [5-11]. Decision trees, however, have not been thoroughly investigated for this application. Of notable exception is the work by Wehenkel et al. [12-14] investigating decision trees for predicting the system's susceptibility to a particular fault. In that work, decision trees take as input the static parameters from a pre-fault operating point and then predict whether the critical clearing time of a particular fault falls below a certain threshold. Control strategies are suggested for moving the operating point to a more secure state [12].

Much of the previous work on pattern recognition has traditionally focused on similar questions of dynamic security, i.e. measuring the system's susceptibility to various contingencies as a function of the operating point. The emerging capability to rapidly acquire synchronized phasor measurements enables us to take a different approach. Using a short window of post-fault phasor measurements from a fault that is actually in progress, we seek to predict whether loss of synchronization is going to occur before it actually happens. This information could then be used for example in out-of-step relaying.

Our method incorporates a combination of features which set this work apart: We simulate a non-trivial power system, the New England 39 bus test system, under increased loading conditions in order to exhibit instances of instability caused by

faults less than 8 cycles in duration (typical breaker failure time). Stability prediction is based on a short (8 cycle) window of realistic-precision, post-fault phasor measurements. We construct a single tree to handle all fault locations in the network, including all the bus faults as well as randomly located line faults. Faults in the test set range from 1 to 8 cycles in duration. Every fault that we simulate is cleared by removing the faulted transmission line (our "bus" faults represent transmission line faults occurring near a bus). The trees are constructed for a particular loading condition, but their robustness to variations in the operating point is investigated using a test set of 40,800 faults from 50 randomly generated operating points. Section 3 provides an overview of the methodology while Section 4 fills in details and presents results.

# 2. DECISION TREES

Decision or classification trees provide an established technique for solving classification problems that have a small number of categories (e.g. stable vs. unstable). Successful applications include medical diagnosis [3], fusion of sensor measurements [15], and assessment of stability margins in electric power systems [12-14]. In this paper we employed one of the most effective and popular methods for building decision trees: the recursive partitioning algorithm outlined in Section 2.2 below [3,16,17].

## 2.1 Background and Terminology

Decision trees are constructed from a training set of examples. Each example in the training set consists of an input vector, along with its correct classification. The tree building process seeks to fit the training set data without over-fitting the data. The resulting tree is tested on an unseen test set where the predicted class is compared with the true class for each example. The classification error rate for the test set measures the method's success.

A decision tree classifies each input vector according to a series of tests. The diagram of a decision tree is a flow chart in the shape of an upside-down tree (Fig. 1). Starting at the top node, the flow branches right or left depending on the outcome of a simple test. For numerical data, the test is whether a particular element of the input vector exceeds a threshold. Processing proceeds down the tree until a terminal node is reached. The input is classified according to the class of the terminal node.

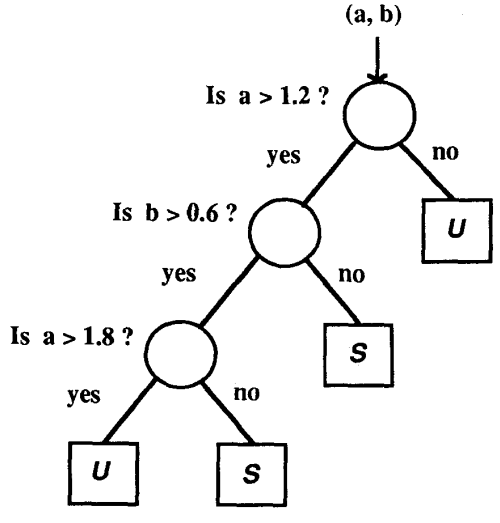



Figure 1: A simple decision tree for illustration purposes.

The implementation of a decision tree is compact and extremely fast. Each processor node is equivalent to an if-then-else block of commands in a high-level computer language. The tree's depth is the maximum number of conditional branches that must be executed before reaching a terminal node. Typical depths are relatively small compared to the capabilities of modern computers. For example the trees in this paper range from 7 to 14 nodes in depth. Since the conditional test at each node is simple, classification requires little computation.

## 2.2 CART Tree-Building Algorithm

The CART tree building algorithm is a statistical method which recursively splits the training set into regions of increasing purity in terms of class membership. Each split corresponds to a node of the tree; hence the tree is grown recursively from the top node down. The recursive partitioning algorithm implemented in the CART software package initially grows a large tree and then determines optimal size by pruning [3]. Other techniques are also available for constructing trees [18,19]. Like most statistical methods, success depends upon the fit between the problem and the method's assumptions. We used CART because it fit our problem well and produced good results.

CART builds decision trees using a greedy, exhaustive search. Starting at the top node, CART checks every split on every variable of the input vector and selects the best splitting criterion as defined by an entropy measure of (successor) node purity. This process is repeated for all subsequent nodes until the nodes are pure or else no further increase in purity is possible. After the full tree is grown CART prunes the tree to avoid over-fitting the data. Pruning reduces the effects of statistical outliers that might cause areas of the feature space to be mis-classified. CART reserves a user-specified portion of the training data to prune the full tree.

# 3. ON-LINE STABILITY PREDICTION

As the increase in electric power demand out-paces the installation of new transmission facilities, power systems are forced to operate with narrower margins of stability. A well designed system should be able to withstand breaker failure scenarios. When the primary circuit breakers fail to clear a fault within their domain of protection, it is standard practice to require 8 cycles to pass before more distant circuit breakers intervene. This delay is necessary to prevent all the nearby circuit breakers from operating simultaneously and dismantling the system. For this reason, it is desirable for the system to withstand faults of 8 cycles or less.

The New England 39 bus test system taken from Pai [20] satisfies the above design criterion relatively well for the nominal loading situation. When the load is increased by 25%, however, system robustness suffers dramatically. On the basis of 1700 random faults on the transmission network, from 1 to 8 cycles in duration, which were cleared by tripping the faulted line, 37% resulted in instability. On-line transient stability prediction becomes more urgent in such a stressed system. We choose to investigate the decision tree method for stability prediction under these circumstances. The resulting decision trees are tested on faults from 1 to 8 cycles in duration in order to cover the range of actual fault clearing times.

## 3.1 One Tree for All Fault Locations

The approach of this paper is to construct a single tree to predict system stability following a three-phase, short circuit to ground fault anywhere in the network. This task is accomplished using synchronized phasor measurements from all the generator busses as input vectors. Three samples, four cycles apart, are taken from each generator - enough to approximate initial velocities and accelerations. We hypothesized, and our results

confirm that the generator angle measurements contain enough information to predict stability in most cases.

A decision tree designed to work for arbitrary fault locations must have a sufficiently diverse training set. Preliminary training sets were obtained by simulating faults of various lengths on all the busses. Although bus faults represent severe disturbances, it is relatively unlikely for a fault to occur on the aluminum grid work of the bus itself. What is meant by a bus fault, realistically speaking, is for a fault to occur on one of the transmission lines feeding into the bus. Mathematically, the two are equivalent because a short section of transmission line has negligible impedance. In the real world, however, line faults are cleared by removing the transmission line, whereas a bus fault would require isolating the bus. This observation motivates the selection of a training set as follows.

The training set contains examples of three-phase, short circuit to ground faults on either end of every transmission line with the transmission line removed at clearing time. Faults of several durations are simulated for each location. Each example contains the post fault phasor measurements, along with whether the fault produced instability. A decision tree is generated to fit the training set, and then tested on new, unseen data. In particular, we test whether the tree performs well on line faults away from the busses. These results are detailed in Section 4.

## 3.2 Robustness to Load Variations

It is important to determine whether a classification scheme can tolerate variations in the operating point. On one hand, it would be reasonable to generate several decision trees to cover the range of total loads. There could be, for example, a tree for 40% of peak load, a tree for 42% of peak load, etc. On the other hand, it would not be practical to build a separate tree for every different combination of individual loads - the dimension of the space is too high. For this reason, we build a tree for a particular value of aggregate load, and investigate its robustness to variations in the individual loads. A test set of 40,800 different faults from 50 randomly generated operating points was generated in order to examine this robustness. Results are given in Section 4.

## 3.3 Robustness to Measurement Imprecision

When investigating a technique for real-world application, care must be taken not to rely on the high precision of digital simulation. We addressed this issue by truncating output from the simulation program before using it for classification. Specifically, the post-fault generator angles were written to a file in units of radians with three digits of precision after the decimal. The velocities and accelerations were computed from this truncated data. Note that 0.001 radians, the precision of our simulated measurements, corresponds to 0.057 degrees. The precision of synchronized angle measurements is determined by the precision of the synchronizing pulse, since the individual phasor estimates are very accurate. The 1 microsecond accuracy available from the GPS satelite transmissions corresponds to 0.0216 electrical degrees at 60 Hertz [21]. Hence our simulated measurement accuracies are realistic.

The timing of post-fault phasor measurements relative to clearing time is another area of imprecision. In the current scheme, each generator provides three samples, four cycles apart beginning immediately after clearing time. Since the synchronization of phasor measurements would be independent of fault occurrences, up to four cycles could pass before the first measurement occurs. It is necessary then, for the classifier to tolerate variations in the sampling start-time. Hence for every fault, we produce one set of phasor measurements where sampling begins at clearing time, and another set where sampling begins two cycles later; both examples are included in the data set.

There is an additional reason for including faults with delayed sampling in the training sets. As explained in Section 2.2, CART randomly withholds one third of the training set in the process of tree building for the purpose of "pruning". Because our training sets are generated systematically rather than randomly, withholding one third of the data could eliminate valuable information. On the other hand, it would defeat the purpose of witholding data to merely duplicate each example in the training set. As a solution, delayed examples are added, thus ensuring adequate representation for all the faults. At the same time, this procedure incorporates robustness to variations in the start-sample time.

# 4. SIMULATIONS

The decision tree method was investigated using the New England 39 bus test system as reported in [20]. In the classical model, each generator is represented as a constant voltage source behind its transient reactance and the loads are constant impedance. The generator angles are governed by the real-power swing equations:

$$\dot{\delta}_i = \omega_i$$
$$M_i\dot{\omega}_i = P_{mi} - \sum_j E_i E_j Y_{ij}\cos(\delta_i - \delta_j - \theta_{ij}) \quad (1)$$

where

| | |
|---|---|
| $\delta_i$, $\omega_i$ | rotor angles and velocities |
| $M_i$ | inertia coefficients |
| $P_{mi}$ | mechanical input powers |
| $E_i$ | generator voltages |
| $Y_{ij}$, $\theta_{ij}$ | admittance magnitudes and angles |

## 4.1 Basic Methodology

Each example of a fault contains the simulated post-fault phasor measurements along with whether the particular fault results in instability. Large numbers of examples are aggregated together into training and test sets, from which trees are constructed and tested. The following sections describe the precise methodology for generating the various training and test sets.

### 4.1.1 The Predictors

Stability prediction is based on an eight cycle window of phasor measurements which begins at fault clearing time, $T_c$. Three consecutive measurements, four cycles apart, are taken from each of the ten generator angles: The first measurement at $T_c$, another at $T_c$ + 4/60, and the last at $T_c$ + 8/60. The generator angles, measured in radians and in center of angle coordinates, are first written to a file in FORTRAN (F8.3) format. This truncates the data to three digits after the decimal.

Two velocities and one acceleration are computed from the truncated generator angles, for a total of six predictors per generator. Denoting the three angle measurements from the i'th generator $\delta_i(0)$, $\delta_i(1)$, $\delta_i(2)$, we compute

$$\begin{aligned} v_i(0) &= 10 * [\delta_i(1) - \delta_i(0)] \\ v_i(1) &= 10 * [\delta_i(2) - \delta_i(1)] \\ a_i(0) &= 20 * [\delta_i(2) - 2 * \delta_i(1) + \delta_i(0)] \end{aligned} \quad (2)$$

Consequently each example contains sixty predictors in FORTRAN (F8.3) format.

Two examples are generated for every fault. For the first example, the eight cycle window of phasor measurements begins exactly at clearing time as described above. For the second example, the data window begins two cycles later. Thus in either case, measurements are completed within 10 cycles of clearing time.

#### 4.1.2 The Predictand

For a given fault location, duration and clearing action, the fault-on and post-fault trajectories are integrated by the fourth-order Runge-Kutte method. The criterion for instability is whether the difference between any two generator angles exceeds 360 degrees in the four seconds after clearing time. Otherwise the fault is declared stable. We found this to be a good criterion for instability in the 39 bus system. Of 172 faults producing pole slip within four seconds: only six oscillated more than two seconds before pole slip occurred, and only one oscillated more than three seconds before pole slip occurred.

#### 4.1.3 Operating Points

Several operating points were generated in order to test the decision tree method on a stressed system, and in order to study the method's robustness to variations in the operating point. Our base case was obtained by increasing the real powers of the individual loads by 25%. The extra power generation was spread uniformly among the generators. We chose an increase of 25% because it lowered critical clearing times, while maintaining an acceptable load-flow solution. Equilibrium voltage magnitudes were relatively unchanged by this increase in loading, but the voltage angles were noticeably different. Specifically, angle differences across transmission lines were larger, reflecting increased amounts of power flow.

Some notation is required in order to conveniently label the various operating points. The case of 25% load increase will be called OP 125 since the loads are 125% of their nominal levels. Hence the operating point specified in the original data is OP 100. Similarly OP 120 and OP 130 have real power load increases of 20% and 30% respectively.

Fifty-five additional operating points were generated by randomly varying the loads. Each individual bus load was randomly assigned a value between 120% and 130% of its nominal level. The distribution of the random numbers was uniform rather than Gaussian, and a different string of random numbers was used for each operating point. For each of these operating points, the increase in generation was distributed evenly among the generators.

#### 4.1.4 Training Sets and Trees

Faults of varying location, duration and clearing action were simulated for each operating point. In every case, the fault type was three-phase short circuit to ground, and the clearing action was transmission line removal. In this discussion, a bus fault refers to a fault on the end of a transmission line which is cleared by removing the line. With 34 transmission lines in the network, there are 68 such scenarios. A mid-line fault refers to a fault in the middle of a transmission line. For the training sets, we simulate a range of fault durations from 1 to 10 cycles. Hence we compute 680 bus faults and 340 mid-line faults per operating point.

Six candidate decision trees are constructed from different combinations of the training data. The data set for Tree 1 contains bus faults from OP 125 only. Tree 2 is trained on bus faults from OP's 120, 125 and 130. Tree 3 is trained on bus faults from OP 125 as well as bus faults from 5 of the randomly generated operating points. Trees 1A, 2A and 3A are the same as 1, 2, and 3 except that mid-line faults were also included. These configurations are summarized in Table I below.

Table I: Composition of training sets for the various trees.

| Tree | Fault Types | OP's | # |
|---|---|---|---|
| 1 | bus | 125 | 680 |
| 1A | bus, mid-line | 125 | 1020 |
| 2 | bus | 120, 125, 130 | 2040 |
| 2A | bus, mid-line | 120, 125, 130 | 3060 |
| 3 | bus | 125, 5 random | 4080 |
| 3A | bus, mid-line | 125, 5 random | 6120 |

#### 4.1.5 Test Sets

Faults of random location and duration were simulated for 50 randomly generated operating points and also for the base case operating point, OP 125. The fifty randomly generated operating points were separate from the five used in the training sets. Bus faults were also computed for the 50 random operating points. Bus faults from OP 125 had already been included in the training sets.

Test Set 1 was obtained from OP 125 as follows. For each transmission line, 50 fault locations and durations were selected at random. The location ranged from 1 to 99 percent of the length of the line, and the duration ranged from 1 to 8 cycles. Test Set 3 comes from the 50 random operating points in the same way, except using 8 faults per line. The number of faults in Test Set 3 is (50 OP's) x (34 lines) x (8 faults/line) = 13,600.

Test Set 2 contains bus faults from the 50 operating points. The 68 scenarios of a fault on either end of every transmission line were simulated with durations of 1,2,...,8 cycles. Table II summarizes the test set information. Again, all faults were cleared by removing the faulted transmission line.

Table II: Composition of the test sets.

| Set | Fault Types | OP's | # |
|---|---|---|---|
| 1 | random - line | 125 | 1700 |
| 2 | bus | 50 random | 27200 |
| 3 | random - line | 50 random | 13600 |

#### 4.1.6 Computational Issues

The generation of training and test sets which occurs in the off-line phase does not present an excessive computational burden. Because each fault is computed independently of the others, parallel implementation is trivial. It is sufficient, for example, to create 5 copies of the program, configure each copy to generate 1/5 of the data, and run the 5 copies on 5 different computers.

Test Set 2 was generated in parallel on a cluster of IBM RISC System/6000's at the Cornell National Supercomputer Facility (CNSF). The Parallel Virtual Machine (PVM) software developed at Oak Ridge National Laboratory was used to automate the process. The PVM subroutine library enables multiple copies of FORTRAN or C programs to run simultaneously on different machines, and to pass messages back and forth. For our application, a short master program initiated multiple copies of the fault simulation program and directed each copy to generate a portion of the data.

The 27,200 faults in Test Set 2 took approximately 2.5 hours of wall clock time using 5 of the RS/6000's in parallel. Due to the presence of other users on the system, different instances of the program would finish faster than others so that some of the 2.5 hours was spent waiting for slower machines to finish. Test Set 3 containing half as many faults required approximately one hour of CPU time (from one of six processors) on the IBM ES/9000 supercomputer at the CNSF. Hence the wall clock time of the parallel implementation roughly corresponds to the CPU time of the supercomputer.

The bulk of our computation was directed toward generating Test Sets 2 and 3, in order to investigate robustness. The training sets required substantially less computation. Tree building was a modest computation for the size of the training sets involved. Even the testing proceeded rather quickly due to the speed of decision tree classification.

Table III: Classification rates for the training data and the OP 125 data.

| Tree | Nodes | Training Set | | Test Set 1 | |
|---|---|---|---|---|---|
| | | Stab | Unst | Stab | Unst |
| 1 | 28 | 97.9 | 98.1 | 97.2 | 91.0 |
| 1A | 26 | 97.4 | 99.1 | 97.1 | 95.5 |
| 2 | 30 | 96.3 | 97.1 | 96.8 | 93.8 |
| 2A | 48 | 97.2 | 96.6 | 96.5 | 92.2 |
| 3 | 55 | 98.1 | 98.9 | 97.5 | 93.5 |
| 3A | 117 | 99.6 | 99.8 | 98.2 | 97.1 |

Table IV: Robustness performance.

| Tree | Nodes | Test Set 2 | | Test Set 3 | |
|---|---|---|---|---|---|
| | | Stab | Unst | Stab | Unst |
| 1 | 28 | 95.8 | 92.2 | 97.7 | 92.5 |
| 1A | 26 | 96.0 | 94.7 | 98.2 | 94.7 |
| 2 | 30 | 96.5 | 94.4 | 97.2 | 95.5 |
| 2A | 48 | 95.1 | 92.5 | 95.4 | 93.9 |
| 3 | 55 | 96.9 | 95.1 | 98.4 | 93.8 |
| 3A | 117 | 95.6 | 95.7 | 96.5 | 95.5 |

## 4.2 Results

Each tree was tested on all three test sets and the results are given in Tables III and IV. The numbers indicate the percentage of inputs correctly classified. Separate percentages are listed for stable vs. unstable cases, i.e. the number in the stable column indicates the percentage of actually stable cases that were correctly classified. Classification rates for the training set are also listed in Table III.

## 4.3 Discussion

The decision trees range in size from 26 to 117 terminal nodes, and the number of nodes generally increases with the size of the training set. A useful fact is that the number of processor nodes is always one less than the number of terminal nodes. The six tree depths are 7, 9, 10, 9, 14 and 14 layers respectively. As previously explained in Section 2, the tree depth bounds the number of conditional tests that are required for each classification. For a depth of 14, actual classification time will be negligible compared to the acquisition of phasor measurements.

The trees attain excellent performance on the training sets which is significant because the training sets include faults from all the busses. If the system experiences a bus fault anywhere in the network, the probability of correctly predicting stability is determined by the classification accuracy for the training set. Tree 3A achieves almost 100% accuracy on both stable and unstable cases in the training set.

Test Sets 1, 2, and 3 were designed to measure the trees' ability to generalize. Test Set 1 contains randomly generated line faults from the base case operating point. For stable cases in Test Set 1, classification accuracy is close to that on the training set. The unstable cases present more difficulty. Trees 1A and 3A, both trained on mid-line faults, perform reasonably well in the unstable category for Test Set 1.

Test Sets 2 and 3, measure the trees' robustness to variations in the operating point. Together these test sets contain 40,800 faults from 50 randomly generated operating points. Test Set 3 is similar to Test Set 1 in that both contain randomly generated line faults, so it is interesting to note the similarity in performance between the two. All the numbers listed under Test Set 3 are within two percentage points of those for Test Set 1, and the differences go both ways. This demonstrates excellent robustness to variations in the load configuration. Test Set 2 also obtains similar results except that accuracies for stable cases are slightly diminished.

### 4.3.1 Observations

On the basis of Test Sets 2 and 3, trees 1 and 2A could be excluded because of their weak performance (92-93%) on unstable cases. Using bus faults from the base case operating point alone (Tree 1) was apparently insufficient; the addition of mid-line faults was beneficial (Tree 1A). For some reason, the performance of Tree 2 was degraded by the addition of mid-line faults (Tree 2A). Recall that Trees 2 and 2A are trained on a fairly wide range of load conditions (+- 5% total load) and it is possible that this range is too large. Note that Trees 2 and 2A have lower scores on the training data.

Tree 3A, with 117 terminal nodes, did not continue to supersede its counterparts when presented with data from the random operating points. Although it still performs well, we believe that Tree 3A has been over-trained. Tree 3A, for example, has the least consistent performance from operating point to operating point. We measured the trees' consistencies by calculating performances for the individual operating points. Means and standard deviations were computed from these numbers. The means were just slightly different from the numbers in Table IV, because different operating points had

different proportions of stable and unstable cases. The standard deviations are reported in Table V below.

Table V: Standard deviations of the observed prediction accuracies for the 50 randomly generated operating points.

| Tree | Nodes | Test Set 2 | | Test Set 3 | |
|---|---|---|---|---|---|
| | | Stab | Unst | Stab | Unst |
| 1 | 28 | 1.3 | 2.0 | 1.1 | 2.3 |
| 1A | 26 | 1.0 | 2.3 | 0.8 | 2.5 |
| 2 | 30 | 0.9 | 1.4 | 0.6 | 1.6 |
| 2A | 48 | 1.4 | 1.0 | 1.7 | 1.8 |
| 3 | 55 | 1.1 | 1.8 | 0.9 | 1.9 |
| 3A | 117 | 2.8 | 2.7 | 2.7 | 3.1 |

Tree 3A has the highest standard deviations in all categories while those for Tree 2 are much lower. The lower the standard deviation, the more consistently the mean performance is achieved from operating point to operating point. Hence there seems to have been a gain in consistency from training Tree 2 on a small range of total loading conditions. Tree 3A had attempted to learn the space of random operating points by training on faults from 5 random operating points. As a result, Tree 3A performs extremely well for some operating points and less well on others, though its overall performance is still fairly good.

#### 4.3.2 Suggestions for Improvement

The above observations can be translated into the following suggestions for improvement. The first is to incorporate a range of total load variation into the training set as in Tree 2, but with a tighter spread of loading range. Try for example training on OP's 123, 124, 125, 126 and 127. Another suggestion is to investigate alternative choices for line faults to include in the training sets. Rather than simply using mid-line faults, one could easily include faults at 25%, 50% and 75% of the length of the line. And another interpretation of the data suggests that it could be beneficial to include line faults from the base case operating point only, since Trees 2A and 3A showed mixed results from the inclusion of line faults.

## 5 FURTHER ISSUES

In any attempt to provide real-time prediction on a system-wide basis, a tradeoff exists between speed and accuracy. A basic limitation is the number of synchronized phasor measurement units (PMU's) that one can afford to install. These units are necessary for measuring the post-fault system state, which along with the governing system equation, determines the ultimate system stability. Even if the complete system model could be solved in real-time, predicting future behavior would still require knowing the system state. However the size of present day power systems vastly exceeds the capability for instantaneously measuring the post-fault system state. Actual numbers of generators typically range in the hundreds, whereas utilities more typically contemplate installing dozens of PMU's [22]. Hence the limited number of PMU's necessitates a reduced-order model. Such a model can be obtained through coherency reduction [23].

Having a reduced-order model is also important regarding the computational burden. If one were intending to solve the model in real-time for a given set of post-fault phasor measurements, then the model would be strictly constrained by the need for real-time solutions. In contrast, pattern recognition approaches, which are based on extensive off-line simulations, provide more latitude for model complexity. The parallel nature of generating training sets extends this flexibility.

Although they have the ability to train off-line, pattern recognition approaches are not entirely immune from the tradeoff between speed and accuracy. An actual implementation would require simulating a large number of faults for a large number of system configurations. This fact will require that a reduced-order model be used. An advantage of pattern recognition, however, is that a more sophisticated reduced-order model can be used off-line.

### 5.1 Accuracy

Potential applications in real-time protection demand high accuracy from the system model. The more traditional task of dynamic security assessment [24] provides a wider margin for error because the question is whether the system is susceptible to a variety of postulated contingencies, and whether preventive control action should be taken. In that context, it is acceptable to provide conservative predictions because at least the postulated contingencies will be protected against, albeit at some economic inefficiency.

The potential applications for real-time stability prediction impose a different set of constraints on accuracy. Real-time stability prediction could be used to trigger "special protection schemes" such as controlled system separation, or tripping unstable generators along with their associated loads. In order to fuel these potential applications, we are concurrently developing a parallel network of decision trees to predict unstable groups of generators [25]. In any case, the fact that real-time prediction would be used for real-time protection schemes will motivate different concerns for accuracy.

Consider, for example, out-of-step relaying. Impedance relays along the Florida-Georgia border have sometimes tripped as a result of large, stable swings caused by loss of generation in Florida [26]. It would be useful, then, to block these relays in the event of a large stable swing (out-of-step blocking). For this application, it would be desirable not to block the relays in the case of an unstable swing. Progress will be achieved, however, if some portion of the false trips are prevented. Hence we should like to use a slightly conservative model for this out-of-step relaying problem.

The balance between conservative and optimistic prediction costs will ultimately depend on the application. For example, mistakenly triggering separation of the WSCC system can be handled fairly routinely. On the other hand, failure to execute special protection schemes where needed can prove quite costly [27]. An advantage of pattern recognition methodology is the flexibility to choose from a range of models between conservative and optimistic.

Some models are well known for giving optimistic results - predicting stability in the case of instability, while others give conservative results. The constant impedance load model generally gives optimistic results, while the constant P-Q load model generally gives conservative results [28]. It has been shown that better generator and load models give more accurate results [29,30].

Any method of performing real-time stability prediction has to rely on some model and its inherent accuracy. This section has thus far addressed the accuracy of the model with respect to the actual system. On one hand, the accuracy of the model is necessarily limited by the availability of phasor measurements and computing resources. On the other hand, the pattern recognition methodology permits greater flexibility in choosing the best model within these constraints.

Our paper has shown that a decision tree is capable of learning a particular model with good accuracy. The classical model with constant impedance loads is overwhelmingly favored for pattern recognition studies because of its accessibility. As indicated earlier, there exists both a need and an opportunity to explore the efficacy of this approach using models of greater sophistication. A logical extension of this work would be to train and/or test decision trees using more sophisticated load and generator models. The structure preserving model, Transient Network Analysis (TNA) models or industry simulation packages could be used in further studies. The idea is to train with a model of sufficient accuracy to predict real-world behavior.

## 5.2 Changing Conditions

Increased model complexity increases the off-line computational requirements of training a pattern recognition technique. Too much computational burden will make it difficult to handle the variety of loadings, system configurations, and generator unit commitments. With a small, though non-trivial model, it would be possible to compute new decision trees on-line as system conditions change. A 1020-fault training set for the 39-bus system requires just a few minutes of wall clock time on the cluster of RS/6000's even with other users on the system. Without other users, we estimate a computation time of about 2 minutes for such a training set. Tree building takes 62 seconds of CPU time on a single RS/6000 for this size training set.

Clearly there is room to compromise between speed and accuracy if the tree for a 10-machine system can be obtained in 3 minutes. In a sense, the decision tree methodology automates the process of deriving relay logic on the basis of off-line studies. Large numbers of detailed simulation outputs can be handled routinely. The rate limiting factor is how quickly training data can be simulated, not how quickly it can be assimilated. This opens exciting possibilities for adaptively changing prediction logic to accommodate new operating configurations. The parallel nature of running multiple simulations, and the potential payoff from system-wide instability detection permit the off-line computational requirements to be met.

# 6. CONCLUSIONS

We have demonstrated the success of properly trained decision trees in predicting transient stability from a short window of post-fault phasor measurements. Extensive testing was performed on the New England 39 bus system under heavy loading conditions. We have shown the adequacy of a single decision tree for all fault locations, with classification accuracies as high as 97-98%. Robustness to variations in the operating point was investigated using a test set of 40,800 faults from 50 randomly generated operating points. Accuracies in excess of 95% were also obtained for these contingencies.

The decision trees were constructed off-line from simulated data. The training sets included faults of various durations on all the busses and all the transmission lines. The computational burden proved to be quite reasonable, and larger systems could be handled. Since individual faults are generated independently, parallel implementation is trivial. Even the larger test sets were easily handled by parallel computation. Once the tree is constructed, the on-line implementation is compact and extremely fast.

We are recommending multiple decision trees to cover the range of loading conditions. The trees' robustness to variations in the operating point determines how many different trees are needed. We investigated the influence of training set composition on robustness performance. We found that consistently good results were achieved by training on faults from a uniform spread of loading conditions. Trees that were trained on faults from randomly generated operating points performed as well on average, but did not possess the same consistency. Simply adding more faults to the training set does not always increase robustness performance. Hence we have outlined specific strategies for incorporating sufficient diversity into the training set while avoiding over-training.

We have argued that a reduced-order model is necessary for any method of predicting transient stability in real-time. With a pattern recognition approach, however, computation occurs off-line which offers greater flexibility in choosing the system model. Since the tradeoff takes place between accuracy and off-line computation, the cost of increased accuracy is reflected in decreased adaptability. We are encouraged, however, by the speed of tree-building for our 10-machine system. This observation suggests the possibility of adaptively recomputing decision trees on-line in response to changing system conditions. We suggest that a decision tree methodology can automate the process of transforming off-line simulation studies into on-line decision rules.

# 7. ACKNOWLEDGEMENTS

This work was partially supported by the National Science Foundation under grant ECS-8913460. Computer results were generated at the Cornell National Supercomputer Facility which is funded in part by the National Science Foundation, New York State, and the IBM Corporation. We are thankful to Bih-Yuan Ku for his programming assistance.

# 8. REFERENCES

[1] A. G. Phadke and J. S. Thorp, "Improved Control and Protection of Power Systems through Synchronized Phasor Measurements", *Control and Dynamic Systems*, Vol. 43, pp 335-376, Academic Press, New York, 1991.

[2] R.P. Schulz, L.S. VanSlyck, and S.H. Horowitz, "Applications of Fast Phasor Measurements on Utility Systems", PICA Proc., pp. 49-55, Seattle, May 1989.

[3] L. Breiman et al., *Classification and Regression Trees*, Wadsworth, Belmont, California, 1984.

[4] S.R. Safavian, and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology," *IEEE Transaction on Systems, Man and Cybernetics*, Vol. 21, No. 3, pp. 660-674, 1991.

[5] C.K. Pang et al., "Security Evaluation in Power Systems Using Pattern Recognition", *IEEE Trans. on Power Apparatus and Systems*, PAS-93, pp. 969-976, 1974.

[6] H. Hakimmashhadi, and G.T. Heydt, "Fast Transient Security Assessment", *IEEE Trans. on Power Apparatus and Systems*, PAS-102, No. 12, pp. 3816-3824, 1983.

[7] S. Yamashiro, "On-Line Secure-Economy Preventive Control of Power Systems by Pattern Recognition", *IEEE Trans. on Power Systems*, PWRS-1, No. 3, pp. 214-219, 1986.

[8] J.A. Pecas Lopes, F.P. Maciel Barbosa, and J.P. Marques De Sa, "On-Line Transient Stability Assessment and Enhancement by Pattern Recognition Techniques", *Electric Machines and Power Systems*, Vol. 15, No. 4-5, pp. 293-310, 1988.

[9] D.J. Sobajic, and Y.H. Pao, "Artificial Neural-Net Based Dynamic Security Assessment for Electric Power Systems", *IEEE Trans. on Power Systems*, PWRS-4, No. 1, pp. 220-228, 1989.

[10] J.L. Souflis, A.V. Machias, and B.C. Papadias, "An Application of Fuzzy Concepts to Transient Stability Evaluation", *IEEE Trans. on Power Systems*, PWRS-4, No. 3, pp. 1003-1009, 1989.

[11] D.R. Ostojic, and G.T. Heydt, "Transient Stability Assessment by Pattern Recognition in the Frequency Domain", *IEEE Trans. on Power Systems*, PWRS-6, No. 1, pp. 231-237, 1991.

[12] L. Wehenkel, Th. Van Cutsem, and M. Ribbens-Pavella, "Decision Trees Applied to On-Line Transient Stability Assessment of Power Systems", *Proc. IEEE Int. Symp. on Circuits and Systems*, Vol. 2, pp. 1887-1890, Espoo, Finland, 1988.

[13] L. Wehenkel, Th. Van Cutsem, and M. Ribbens-Pavella, "An Artificial Intelligence Framework for On-Line Transient Stability Assessment of Power Systems", *IEEE Transactions on Power Systems*, PWRS-4, No. 2, pp. 789-800, 1989.

[14] L. Wehenkel, M. Pavella, "Decision Trees and Transient Stability of Electric Power Systems", *Automatica*, Vol. 27, No. 1, pp. 115-134, 1991.

[15] D.E. Brown, V. Corruble, and C.L. Pittard, "A Comparison of Decision Tree Classifiers with Neural Networks for Multi-Modal Classification Problems", *Pattern Recognition*, 1993, forthcoming.

[16] E.B. Hunt, J. Marin, and P.J. Stone, *Experiments in Induction*, Academic Press, New York, 1966.

[17] J.H. Friedman, "A Recursive Partitioning Decision Rule for Nonparametric Classification", *IEEE Transaction on Computers*, C-26, pp. 404-408, 1977.

[18] I.K. Sethi, and G.P.R. Sarvarayudu, "Hierarchical Classifier Design Using Mutual Information", *IEEE Trans. on PAMI*, PAMI-4, no. 4, pp. 441-445, 1982.

[19] W.Y. Loh, and N. Vanichsetakul, "Tree Structured Classification Via Generalized Discriminant Analysis (With Discussion)", *J. Am. Stat. Assoc.*, Vol 83, pp. 715-728, 1988.

[20] M.A. Pai, *Energy Function Analysis for Power System Stability*, Kluwer, Boston, 1989.

[21] A.G. Phadke et al., "Synchronized Sampling and Phasor Measurements for Relaying and Control", IEEE PES Winter Meeting, Columbus, Ohio, February 1993 (93 WM 039-8-PWRD).

[22] A.G. Phadke, "Synchronized Phasor Measurements in Power Systems", *IEEE Computer Applications in Power*, Vol. 6, No. 2, pp. 10-15, 1993.

[23] J.C. Giri, "Coherency Reduction in the EPRI Stability Program", *IEEE Trans. on Power Apparatus and Systems*, PAS-102, No. 5, pp. 1285-1293, 1983.

[24] A.A. Fouad et al., "Dynamic Security Assessment Practices in North America", *IEEE Trans. on Power Systems*, PWRS-3, No. 3, pp. 1310-1321, 1988.

[25] S.E. Kretsinger, S.M. Rovnyak, D.E. Brown, and J.S. Thorp, "Parallel Decision Trees for the Real-Time Prediction of Synchronized Groups of Unstable Electric Generators", Technical Report IPC-TR-93-002, Institute for Parallel Computation (present phone 804-924-1043), School of Engineering and Applied Science, University of Virginia, May 10, 1993.

[26] A.A. Fouad et al., "Investigation of Loss of Generation Disturbances in the Florida Power and Light Company Network by the Transient Energy Function Method", *IEEE Trans. on Power Systems*, PWRS-1, No. 3, pp. 60-66, 1986.

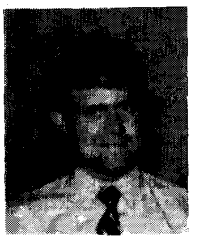
[27] North American Electric Reliability Council, "1988 System Disturbances", July, 1989.

[28] M.H. Kent et al., "Dynamic Modeling of Loads in Stability Studies", *IEEE Trans. on Power Apparatus and Systems*, PAS-88, No. 5, pp. 756-763, May 1969.

[29] E. Vaahedi et al., "Load Models for Large-Scale Stability Studies from End-User Consumption", *IEEE Trans. on Power Systems*, PWRS-2, No. 4, pp. 864-872, 1987.

[30] M.R. Brickell, "Simulation of Staged Tests in the Ontario Hydro Northwestern Region", PICA Proc., pp. 357-364, Cleveland, Ohio, 1979.


**Steven M. Rovnyak** was born in Lafayette, Indiana on July 4, 1966. He received the B.S. degree in electrical engineering and the A.B. degree in mathematics from Cornell University, Ithaca, NY in 1988. He received the M.S. degree in electrical engineering from Cornell University in 1990. Between 1986 and 1988 he spent summers and a fall term researching optical computing and neural networks at the BDM Corporation, McLean, VA. He is presently a graduate student at Cornell University pursuing the Ph.D. degree in electrical engineering. Mr. Rovnyak is a member of Phi Beta Kappa and Phi Kappa Phi. He is a student member of the IEEE.



**Stein E. Kretsinger** was born in Geneva, Switzerland on February 12, 1967. He received the B.A. degree in economics from the University of Virginia in 1989. He is presently a graduate student in the Department of Systems Engineering at the University of Virginia.

**James S. Thorp** (S'58-M'63-SM'80-F 89) received the B.E.E, M.S., and Ph.D. degrees from Cornell University, Ithaca, NY. He joined the faculty at Cornell in 1962 where he is currently a Professor of Electrical Engineering. In 1976 he was a Faculty Intern at the American Electric Power Service Corporation. He was an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS from 1985 to 1987. In 1988 he was an Overseas Fellow at Churchill College, Cambridge, England. He is a member of the IEEE Power System Relaying Committee, CIGRE, Eta Kappa Nu, Tau Beta Pi, and Sigma Xi.

**Donald E. Brown** was born in Panama, CZ on November 1, 1951. He graduated from the United States Military Academy, West Point, with the B.S. degree in 1973. He received the M.S. and M.Eng. degrees from the University of California, Berkeley in 1979 and the Ph.D. degree from the University of Michigan, Ann Arbor in 1986.

He has served as an Officer in the U.S. Army and has worked for Vector Research, Inc. He is currently an Associate Professor of Systems Engineering and Associate Director of the Institute for Parallel Computation at the University of Virginia. His research interests include statistical decision theory and pattern recognition, inductive modeling, and machine learning.

He serves on the Administrative committee of the IEEE Neural Networks Council. He is secretary of the IEEE Systems, Man, and Cybernetics Society and is a former member of the administrative committee of the SMC. He is past-Chairman of the Operations Research Society of America Technical Section on Artificial Intelligence. He is a member of the Pattern Recognition Society of America, the Institute of Industrial Engineers, and a Senior Member of the IEEE.

## Discussion

**L. Wehenkel** (University of Liège, Liège, Belgium): The authors are to be commended for their valuable work on decision trees for transient stability prediction using postfault phasor measurements.

A quite similar idea has been explored for multicontingency voltage security emergency state detection, on the basis of system measurements obtained in the intermediate "just after disturbance state" [A1, A2]. In the latter work, a single decision tree is however designed so as to handle a broad range of variable prefault system configurations (i.e., with variable topology as well as variable load and generation schedules) and a set of disturbances. This allows to build the decision trees off-line, when the actual on-line system configuration is still unknown. Further, it enables one also to classify postfault situations resulting from a cascade of two or more outages. Admittedly, most power systems are designed and also operated so as to withstand at least all single contingencies; thus, the actually dangerous situations generally result from unforeseen coincidences of multiple events. The authors comments on how this problem may be realistically handled in their framework are highly appreciated.

In comparison to other "nonparametric" pattern recognition methods (e.g., nearest neighbor and neural networks), an important strength of the decision tree approach comes from the explicit and easily interpretable classifier that it provides. In the context of power system *preventive* transient stability assessment, this feature has already shown to be of paramount importance for the practical acceptance of the method. It was found, for example, that the information contained in the decision trees may be compared to existing prior expertize, and help to systematically identify the major system weaknesses in terms of its most important attributes [A3]. Could the authors expand on the reasons that made them prefer the decision tree approach to the above quoted competing techniques? Did they find the data analysis feature potentially useful in the context of their problem, or was it simply that the decision trees provided more reliable classifiers than the other techniques?

### References


[A1] T. Van Cutsem, L. Wehenkel, M. Pavella, B. Heilbronn, and M. Goubin. Decision trees for detecting emergency voltage conditions. In *Procs. of the 2nd Int. NSF Workshop on Bulk Power System Voltage Phenomena—Voltage Stability and Security*, Deep Creek Lake, MA, pp. 229–240, Aug. 1991.

[A2] T. Van Cutsem, L. Wehenkel, M. Pavella, B. Heilbronn, and M. Goubin. Decision tree approaches for voltage security assessment. *IEE Proceedings—Part C.*, Vol. 140, no. 3, pp. 189–198, May 1993.

[A3] L. Wehenkel, M. Pavella, E. Euxibie, and B. Heilbronn. Decision tree based transient stability assessment—a case study. *IEEE PES Winter Meeting*, Paper #93 WM 235-2 PWRS, Feb. 1993.


Manuscript received August 16, 1993.

**S. Rovnyak, S. Kretsinger, J. Thorp, D. Brown.** We very much appreciate the comments and references given by Dr. Wehenkel, who has pioneered the application of decision trees in the area of electric power systems stability. In response to his question on our selection of decision trees, it would be fair to say that decision trees were chosen for the same reasons that make them desirable in an actual implementation. Namely that decision trees are accessible and reliable, and have good performance characteristics. The decision trees in this paper were constructed with the aid of a standard software package, using the default settings. After we formulated the method of training set generation, the CART tree–building algorithm consistently achieved excellent classification error rates. Training and testing speed was fast, which is remarkable for a problem having such a high–dimensional input space combined with a large number of cases. These characteristics which proved so valuable in research are essential for the intended application.

Investigations of neural networks seem to indicate that comparable classification error rates are achievable, although training is much slower. These findings are briefly summarized below. Training was performed using a highly optimized gradient descent algorithm designed specifically for this application. Whereas most backpropagation algorithms use Euler's method for computing the gradient descent trajectory, this program utilizes fourth–order Runge–Kutta. The stepsize varies adaptively in order to seek the greatest rate of error reduction. The combination of Runge–Kutta, which permits larger stepsizes, together with an adaptive stepsize produces very rapid training. Furthermore, the program was written to enable vectorization on the ES/9000 supercomputer which speeds execution by a factor of 3.6. As an additional feature, the algorithm escapes from local minima and usually achieves a lower value of error.

This neural network training algorithm clobbered smaller test problems, yet failed to train on the transient stability prediction problem due to the large number of cases and input variables. In order to proceed with the comparison, we eliminated those input variables which had not been utilized by the corresponding decision tree. For instance, Tree 1A in this paper only used 16 of the 60 input variables, and so these were selected as inputs to the neural network. The network

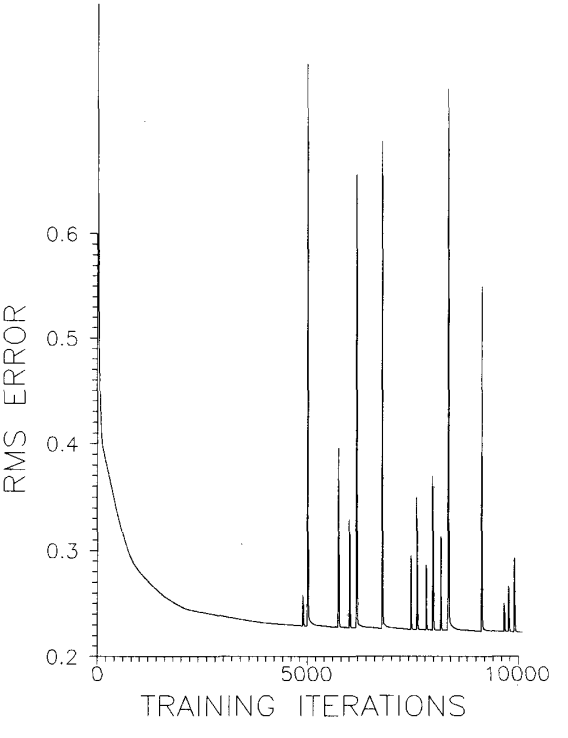



Figure B1: Output error RMS–averaged over the training set.

was trained to associate the stable cases with the value one, and the unstable cases with the value zero. A case is declared stable if the output exceeds 0.5, and unstable otherwise. A slightly larger threshold reduces misclassifications for the unstable cases and increases errors among the stable cases. This feature is useful for balancing the two types of errors.

A network with 10 nodes in a single hidden layer was constructed from Training Set 1A in this paper. With 2040 cases (2 x 1020 faults) having 16 input variables each, the optimized gradient descent algorithm required approximately one hour of CPU time on the ES/9000 supercomputer for 10,000 iterations. The corresponding RMS–averaged output error is shown in Figure B1. The error need not be zero since the output is thresholded prior to classification. Figure B1 shows that most of the error reduction occurs in the first 5,000 iterations, before running into local minima. After 10,000 iterations it appears that further reduction in error will not be achieved.

In order to compute the classification error rate, the output for each case is thresholded at a value close to 0.5. Through experimentation we found that a threshold value of 0.55 produced roughly equal classification error rates among stable and unstable cases. These percentages are given in Tables BI and BII below, along with the classification error rates for Decision Tree 1A. A threshold of 0.52 gives performance characteristics very similar to those of Decision Tree 1A.

Table BI: Classification error rates for training and test data from the base case operating point.

| Classifier Design | Training Set | | Test Set 1 | |
|---|---|---|---|---|
| | Stab | Unst | Stab | Unst |
| Decision Tree | 97.4 | 99.1 | 97.1 | 95.5 |
| NN : Th=.55 | 94.7 | 97.9 | 95.9 | 95.4 |
| NN : Th=.52 | 95.6 | 96.4 | 96.8 | 93.4 |

Table BII: Robustness results – classification error rates for randomly generated operating points.

| Classifier Design | Test Set 2 | | Test Set 3 | |
|---|---|---|---|---|
| | Stab | Unst | Stab | Unst |
| Decision Tree | 96.0 | 94.7 | 98.2 | 94.7 |
| NN : Th=.55 | 95.4 | 95.3 | 96.8 | 96.6 |
| NN : Th=.52 | 96.8 | 93.5 | 98.2 | 95.1 |

The observed similarity in performance accords with previous comparisons of decision trees and neural networks [B1]. In order to train the neural network, however, we had to perform feature selection by first building a decision tree and then determining which variables had been used. Without this reduction of input variables, training by backpropagation did not converge for a reasonable number of iterations. Even with the advantage of feature selection, our optimized neural network algorithm required an enormous amount of computer time. As we have argued in the paper, fast training time is required for adapting to changing system conditions. In order for neural networks to compete with decision trees in this application, faster training would have to be accomplished.

## References

[B1] L. Atlas et al.,"Performance Comparisons Between Backpropagation Networks and Classification Trees on Three Real–World Applications", *Advances in Neural Information Processing Systems*, Vol. 2, pp. 622–629, Morgan Kaufmann Publishers, San Mateo, CA, 1990.

Manuscript received September 30, 1993.